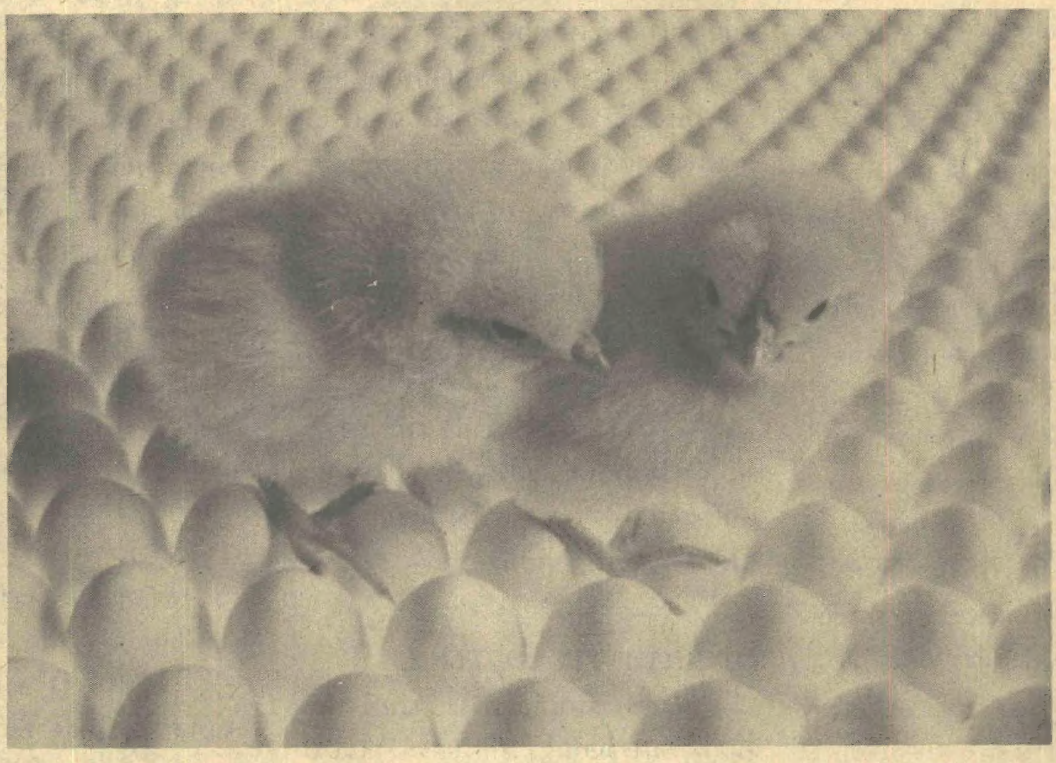

สุนทร วิทูสุรพจนี

เขียนฟังก์ชันจัดการภาษาไทยด้วยภาษาซี

ท่านผู้อ่านมีความคิดเหมือนกับผมไหม ว่า เครื่องไมโครที่มีใช้กันในเมืองไทยเกือบร้อยละ 80 ได้ดัดแปลงกันให้ใช้ภาษาไทย 25 บรรทัดกันได้หมดแล้ว แค่เสียเงินไม่เท่าไร ซื้อการ์ดมาเสียบปั๊บก็เรียบร้อยแล้ว ใช้โปร-แกรมภาษาไทยกันเพลินไปเลย ที่นี้บังเอิญว่าเจ้าเครื่องที่ผมใช้อยู่มาอยู่ในจำพวกอีกร้อยละ 20 ที่ไม่ได้ซื้อการ์ดภาษาไทยมาใส่ เพราะว่าไม่ค่อยได้ใช้ภาษาไทยกันบ่อยนัก จะมีภาษาไทยก็เป็นเพียงตัวอักษรที่เก็บอยู่ใน "รอมกราฟิก" บนเครื่องเท่านั้นแหละ ปัญหาก็เลยเกิดขึ้นมา เมื่อผมต้องการเขียนโปรแกรมใช้งานด้วยภาษาซีและจะต้องให้สามารถแสดงผลกับรับข้อมูลที่เป็นภาษาไทยได้ด้วย ผมก็เลยจำเป็นต้องนั่งเขียนฟังก์ชันจัดการเกี่ยวกับการรับ/ส่งภาษาไทยเพิ่มเข้า

ไปในโปรแกรมด้วย ซึ่งหากว่ามีการ์ดภาษาไทยก็คงไม่ต้องลงทุนถึงขนาดนี้ แต่อย่างไรก็ตาม ผมเห็นว่ามันใช้งานได้ดี ก็เลยอยากจะนำมาเสนอให้คุณ ๆ ได้อ่านกัน ไม่อยากเก็บเอาไว้คนเดียว เรียกว่า "ใจกว้าง" อย่างนั้นแหละ

ทำไมต้องเป็นภาษาซีด้วย

ครับด้วยเหตุว่าโปรแกรมใช้งานของผมเขียนด้วยภาษาซี ก็เลยเขียนฟังก์ชันจัดการเรียงภาษาไทยเป็นภาษาซีด้วย จะได้เข้าชุดกันเท่านั้นเองครับ ความจริงแล้ว ภาษาซีก็สามารถที่จะเชื่อมโยงกับรูทีนที่เป็นภาษาเครื่อง ซึ่งคอมพายล์มาจากภาษาแอสแซมบลีได้ แต่ในที่นี้ผมไม่อยากจะใช้ครับ เพราะหากใช้ภาษาซีซึ่งเป็นภาษาชั้นสูงเขียนแล้ว ตอนดีบักหาข้อผิดพลาดนั้นทำได้ง่ายกว่ากันมากครับ และอีกประการหนึ่งผมจะได้ใช้แนวเขียนดัดแปลงไปเป็นภาษาปาสคาลได้อีกด้วย

แนวคิดเบื้องต้น

ภาษาซีมีคำสั่งที่เกี่ยวข้องกับการรับอินพุทและเอาท์พุทอยู่หลายคำสั่ง เช่น putchar, getchar, printf เป็นต้น ผมต้องการที่จะสร้างฟังก์ชัน (ในภาษาซี ใช้คำว่าฟังก์ชันแทน procedure หรือรูทีนย่อย) เลียนแบบการทำงานของคำสั่งต่าง ๆ เหล่านี้ แต่มีข้อพิเศษอยู่ที่ว่าจะต้องรับข้อมูลภาษาไทย และจัดการได้อย่างถูกต้องเช่นเดียวกับภาษาอังกฤษด้วย ซึ่งผมก็ได้เลือกทำเพียงบางคำสั่งเท่านั้น เพราะเห็นว่าเป็นการพอเพียงแล้ว ฟังก์ชันเหล่านี้ คือ

1. Tgetchar ฟังก์ชันนี้สามารถทำงานได้ลักษณะเดียวกับฟังก์ชัน "getchar" ซึ่งทำหน้าที่รับอักษรมา 1 ตัว โดย Tgetchar สามารถรับอักษรภาษาไทยได้ การกดปุ่ม 'ESC' จะทำให้เปลี่ยนโหมดระหว่างไทยและอังกฤษ และหลังจากกดปุ่มนี้แล้ว ฟังก์ชันยังคงจะรอรับค่าถัดไปอยู่ เพราะถือว่าปุ่มนี้ทำหน้าที่เป็นเพียงการสลับโหมดเท่านั้น

2. Tputchar ฟังก์ชันนี้จะทำการส่งอักษร 1 ตัว ให้ปรากฏบนจอภาพทั้งไทยและอังกฤษ คล้ายกับคำสั่ง "putchar" โดยจะปรากฏอักษรให้อยู่ในลักษณะที่ถูกต้องด้วย

3. Tgets เป็นฟังก์ชันที่จะทำการรอรับอักษรที่อินพุททางคีย์บอร์ด เข้าไปเก็บไว้ในหน่วยความจำที่เป็นบัฟเฟอร์ จากนั้นเมื่อสิ้นสุดการอินพุทหรือครบจำนวนที่ได้กำหนดไว้ ฟังก์ชันจึงจะ return ค่านั้นกลับไปอยู่ในรูปของสตริง ปัญหาสำหรับฟังก์ชันนี้จะยุ่งตรงข้อมูลภาษาไทย เพราะจะต้องตรวจสอบความถูกต้องของการแสดงผลบนจอภาพและการใส่ข้อมูล ให้เป็นไปตามหลักการเขียนภาษาไทยที่ถูกต้องด้วย การลบข้อมูลจะต้องให้ปรากฏให้ถูกต้องด้วยเช่นกัน

4. Tputs เป็นฟังก์ชันที่ทำการส่งข้อมูลที่เป็นสตริง ทั้งภาษาไทยและอังกฤษให้ปรากฏบนจอภาพ การทำงานเช่นเดียวกับฟังก์ชัน "puts" นั่นเอง

โปรแกรม THAIUTIL.C

ฟังก์ชันที่ผมต้องการใช้งานทั้งหมดนั้น ได้นำมารวมกันไว้ในโปรแกรม "THAIUTIL.C" ดังรูปที่ 1 ซึ่งเมื่อผมนำไปใช้งานก็เพียงแต่นำโปรแกรมนี้มาลิงค์เข้ากับโปรแกรมอื่นเท่านี้ก็เป็นอันว่าเรียบร้อย การเขียนและออกแบบโปรแกรมนั้น ผมใช้หลักการเชื่อมโยงแบบเป็นโมดูล (module) คือการที่ฟังก์ชันหนึ่ง ๆ จะทำหน้าที่ได้นั้น จะต้องนำเอาฟังก์ชันย่อยซึ่งอาจจะมีหลายฟังก์ชันมาประกอบกัน วิธีนี้จะทำให้โปรแกรมมีความสวยงาม ง่ายต่อการทำความเข้าใจและตรวจสอบบักด้วย ต่อไปผมจะได้ทำการอธิบายโปรแกรมเป็นส่วน ตามลำดับลงมา

ตอนเริ่มต้นของโปรแกรม (รูปที่ 1) เป็นการ include file "stdio.h" และ "ctype.h" ทั้งนี้เพราะหลาย ๆ ฟังก์ชันได้มีการกำหนดไว้ภายในไฟล์เหล่านี้ ตอนนี้จะเหมือนกับการเริ่มต้นของโปรแกรมที่เขียนด้วยภาษาซีทั่ว ๆ ไป

ลำดับถัดมา เป็นช่วงของการกำหนด (define) ค่าคงที่ต่าง ๆ หลาย ๆ ค่า ซึ่งจุดมุ่งหมายก็เพื่อต้องการที่จะให้โปรแกรมง่ายต่อการทำความเข้าใจ ยกตัวอย่างเช่น Bell มีค่าแทนด้วยเลขฐานสิบหก 07 สมมติว่าเราได้พบคำสั่ง putchar (Bell) เราก็จะรู้ว่าเมื่อทำการส่งค่านี้ขึ้นบนจอภาพ ก็จะได้ยินเสียงปี๊บ (Bell) ดังขึ้นด้วย ซึ่งจะสื่อความหมายได้ดีกว่า putchar (07) ซึ่งก็จะไม่ทราบได้เลยว่า เมื่อทำคำสั่งนี้แล้วจะเกิดผลอย่างใดบ้าง เป็นต้น นอกจากนี้การกำหนดค่าจะทำให้สะดวกยิ่งขึ้น เมื่อต้องการเปลี่ยนแปลงค่าของตัวคงที่นั้น ๆ เข้าไปเพียงแห่งเดียว โดยไม่ต้องตามไปแก้ไขส่วนอื่น ๆ เลย

รหัสภาษาไทย

สำหรับตัวแปรที่เป็นอะเรย์ที่มีชื่อว่า Thaitable นั้น จะเป็นรหัสของค่าตัวอักษรที่สามารถอินพุทได้ในขณะที่อยู่ในโหมดภาษาไทย สำหรับการเรียงลำดับค่านั้น ใช้การจัดตามตำแหน่งค่าของคีย์บอร์ดแบบภาษาไทย ลักษณะเดียวกับแบบที่ใช้ในเครื่องพิมพ์ดีดปกติ

เนื่องจากว่าโปรแกรมของผมในที่นี้ใช้รหัสภาษาไทยตามแบบของ ม.เกษตร–ศาสตร์ ดังรูปที่ 2 จึงมีการกำหนดค่าตามที่เห็นในโปรแกรม หากว่าเกิดต้องการที่จะเปลี่ยนรหัสภาษาไทยเป็นแบบอื่น ๆ การกำหนดค่าก็จะแตกต่างไปบ้าง ดังนั้นผมจึงนำเอาค่ารหัสภาษาไทยหรือที่ผมใช้ว่า Thaitable นี้ กำหนดไว้ในส่วนต้น ๆ ของโปรแกรม เพื่อที่จะได้แก้ไขได้สะดวก ไม่เสียเวลาในการค้นหามากนัก

ตัวแปรในส่วนถัดมาเป็นอะเรย์ต่าง ๆ เช่น Eiarray จะเป็นค่ารหัสอักษรที่เป็นชุดของสระอิ ได้แก่ ◌ิ ◌ี ◌ึ ◌ื และ ◌์ Eeearray เป็นรหัสชุดอักษรที่เป็นชุดของ ◌ี ได้แก่ ◌ี◌่ ◌ี◌้ ◌ี◌๊ ◌ี◌๋ เป็นต้น และนี่เป็นเหตุผลเดียวกัน ที่ผมจัดกลุ่มรหัสเหล่านี้เป็นชุด ๆ ซึ่งแทนด้วยอะเรย์ ทั้งนี้ก็เพียงว่า หากต้องดัดแปลงใช้กับรหัสภาษาไทยแบบอื่น ก็จะแก้ไขเฉพาะส่วนที่เป็นค่าในอะเรย์เท่านั้น ไม่ต้องยุ่งเกี่ยวกับตัวโปรแกรมเลย

ฟังก์ชันย่อยต่าง ๆ

ฟังก์ชันส่วนมากของผม จะมีการใช้การทำงานร่วมกันของฟังก์ชันย่อย ๆ ทั้งนี้

รูปที่ 1 โปรแกรม THAIUTIL.C

```
/* ========================================
 =  Program   : ThaiUtil.c        =
 =-----------------------------------------=
 =  Programer : Suntorn Witosurapot =
 =  Date      : 12 May 1986        =
 ========================================
*/

#include      <stdio.h>
#include      <ctype.h>

#define TRUE  1
#define FALSE 0
#define VOID  int
#define BOOL  int
#define ESC   0x1B
#define CR    0x0D
#define LF    0x0A
#define BS    0x08
#define Bell  0x07
#define Blnk  0x20

#define Ei    0xd9
#define Eee   0xda
#define Eu    0xdb
#define Ewu   0xdc
#define Han   0xdd
#define Amm   0xde
#define Ake   0xe0
#define Tho   0xe1
#define Tri   0xe2
#define Chatawa  0xe3
#define Karun 0xe4
#define Sarau 0xd7
#define Sarawu  0xd8

int Thmode = FALSE;
char Thaitable[] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0xcb, 0x17,
                0x1b, 0x19, 0x1a, 0xae, 0x1c, 0x1d, 0x1e, 0x1f,
                0x20, 0x23, 0x2e, 0x32, 0x33, 0x34, 0xeb, 0xa5,
                0x36, 0x37, 0x35, 0x39, 0xbf, 0xa2, 0xd3, 0xba,
                0xa6, 0x2a, 0x2f, 0x5f, 0xbe, 0xb4, 0xd7, 0xdb,
                0xa3, 0xb3, 0xa9, 0xc4, 0xb0, 0xa8, 0xc9, 0x24,
                0x31, 0xc2, 0x2e, 0xa7, 0xad, 0xac, 0xd2, 0xaa,
                0xdf, 0xb1, 0xe3, 0xc6, 0xc5, 0x3f, 0xe4, 0xd6,
                0xab, 0x30, 0xaf, 0xa4, 0xb6, 0xe2, 0xcb, 0x22,
                0x29, 0xde, 0x28, 0xb8, 0x5c, 0xc3, 0xd8, 0x38,
                0x2d, 0xbd, 0xd9, 0xd1, 0xa1, 0xcf, 0xb2, 0xd0,
                0xe1, 0xc1, 0xe0, 0xce, 0xc7, 0xb5, 0xdc, 0xb7,
                0xc0, 0xd5, 0xbc, 0xc8, 0xcc, 0xda, 0xca, 0xd4,
                0xb9, 0xdd, 0xba, 0xae, 0x7c, 0x2c, 0x25, 0x7f };

char EiArray [] = {0xee, 0xef, 0xf0, 0xf1, 0xf2};
char EeeArray [] = {0xf3, 0xf4, 0xf5, 0xf6};
char EuArray [] = {0xf7, 0xf8, 0xf9, 0xfA};
char EwuArray [] = {0xfB, 0xfC, 0xfD, 0xfe};
char HanArray [] = {0xea, 0xeb, 0xec, 0xed};
char AmmArray [] = {0xe6, 0xe7, 0xe8, 0xe9};


int Tgetchar ()
{
  int Ch = 0;

  do {
    Ch = getch();
    if (Ch == ESC)
      Thmode = (!Thmode ? TRUE : FALSE);
  } while (Ch == ESC);
  return (Thmode ? Thaitable[Ch] : Ch);
}

VOID Tputchar(Ch)
int Ch;
{
  if ((Ch <= 0x7E) || (Ch <= 0xD6))
    putch(Ch);
  else if ((Ch == 0xD7) || (Ch == 0xD8))
    Downputch(Ch);
  else if ((Ch <= 0xFF))
```

```
 83:     Upputch(Ch);
 84: }
 85:
 86: VOID tputs(s)
 87: char    s[];
 88: {
 89:     int   i;
 90:
 91:     for (i = 0; s[i] != '\0'; i++)
 92:       Tputchar (codetch(s,i));
 93: }
 94:
 95: char tgets(s,len)
 96: char s[];
 97: int  len;
 98: {
 99:    int c, i, k;
100:
101:    i = 0;
102:    k = len;
103:    Thmode = FALSE;
104:
105:    while (k <= len && --k > 0 && (c=Tgetchar()) != '\r') {
106:      s[i] = c;
107:      if (c == BS) {
108:        if (i == 0) {
109:          putch(Bell); k++;
110:        }
111:        else {
112:          delchar(s,i);
113:          i--; k += 2;
114:        }
115:      }
116:      else if (checkinp(s,i) == TRUE) {
117:        Tputchar(codetch(s,i));
118:        i++;
119:      }
120:      else {
121:        putch(Bell); k++;
122:      }
123:    }
124:    if (c == '\r')
125:      s[i++] = c;
126:    s[i] = '\0';
127: }
128:
129: VOID Upcursor ()
130: {
131:     putch (ESC); putch (0x5b); putch (0x41);
132: }
133:
134: VOID Downcursor ()
135: {
136:     putch (ESC); putch (0x5b); putch (0x42);
137: }
138:
139: VOID Upputch (Ch)
140: char Ch;
141: {
142:     Upcursor (); putch (BS); putch (Ch); Downcursor ();
143: }
144:
145: VOID Downputch (Ch)
146: char Ch;
147: {
148:     Downcursor (); putch (BS); putch (Ch); Upcursor ();
149: }
150:
151: VOID TReturn ()
152: {
153:     putch (CR); putch (LF); putch (LF); putch (LF);
154: }
155:
156: VOID delup()
157: {
158:    Upcursor(); putch(BS); putch(Blnk); putch(Blnk);
159:    Downcursor(); putch(BS);
160: }
161:
162: VOID deldown()
163: {
164:    Downcursor();putch(BS); putch(Blnk);putch(Blnk);
165:    Upcursor(); putch(BS);
166: }
167:
168: VOID delnorm()
169: {
170:    putch(BS); putch(Blnk); putch(BS);
171: }
172:
173:
174: BOOL istmidle(ch)
175: char   ch;
176: {
```

```
177:    return((ch >= 0x00 && ch <= 0xd6) ? TRUE : FALSE);
178: }
179:
180:
181: BOOL istuper(ch)
182: char    ch;
183: {
184:    return((ch >= 0xd9 && ch <=0xdf) ? TRUE : FALSE);
185: }
186:
187: BOOL iswanyuk(ch)
188: char     ch;
189: {
190:    return((ch == Ake || ch == Tho || ch == Tri ||
191:          ch == Chatawa || ch == Karun) ? TRUE : FALSE);
192: }
193:
194: BOOL istlwer(ch)
195: char    ch;
196: {
197:    return((ch == Sarau || ch == Sarawu) ? TRUE : FALSE);
198: }
199:
200: int codetch(s,i)
201: char    s[];
202: int     i;
203: {
204:    int   ch,old;
205:
206:    ch = s[i];
207:    if (i != 0) {
208:      old = s[i-1];
209:      if (iswanyuk(ch) == TRUE) {
210:        if (istuper(old)  == TRUE)
211:          switch(old) {
212:            case Ei:  ch = EiArray[ch - Ake];
213:                  break;
214:            case Eee: ch = EeeArray[ch - Ake];
215:                  break;
216:            case Eu:  ch = EuArray[ch - Ake];
217:                  break;
218:            case Ewu: ch = EwuArray[ch - Ake];
219:                  break;
220:            case Han: ch = HanArray[ch - Ake];
221:                  break;
222:            case Amm: ch = AmmArray[ch - Ake];
223:                  break;
```

▶

```
224:          }
225:       }
226:    }
227:    return (ch);
228: }
229:
230: VOID delchar(s,i)
231: char s[];
232: int  i;
233: {
234:       if (istmidle(s[i-1]) == TRUE)
235:         delnorm();
236:       else if (istlwer(s[i-1]) == TRUE)
237:         deldown();
238:       else {
239:          if (i-2 >= 0 && istuper(s[i-2]) == TRUE){
240:            delup();
241:            tputchar(s[i-2]);
242:          }
243:          else delup();
244:       }
245: }
246:
247: BOOL checkinp(s,i)
248: char    s[];
249: int     i;
250: {
251:    int   k;
252:
253:    if (istmidle (s[i]) == FALSE) {
254:       if (i == 0)
255:         k = FALSE;
256:       else if (iswanyuk(s[i]) == TRUE && iswanyuk(s[i-1]) == TRUE)
257:         k = FALSE;
258:       else if (istuper(s[i]) == TRUE && istuper(s[i-1]) == TRUE)
259:         k = FALSE;
260:       else if (istlwer(s[i]) == TRUE && istlwer(s[i-1]) == TRUE)
261:         k = FALSE;
262:       else if (istuper(s[i]) == TRUE && iswanyuk(s[i-1]) == TRUE)
263:         k = FALSE;
264:       else
265:         k = TRUE;
266:    }
267:    else  (k = TRUE);
268:    return(k);
269: }
270:
```

4 บิทต้น / 4 บิทท้าย	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø			SP	Ø	@	P		p			◌	ฒ	ย	เ	◌	◌
1			!	1	A	Q	a	q			ก	ณ	ร	แ	◌	◌
2			"	2	B	R	b	r			ข	ด	ฤ	ไ	◌	◌
3			#	3	C	S	c	s			ค	ต	ล	ใ	◌	◌
4			£	4	D	T	d	t			ฆ	ถ	ว	ไ	◌	◌
5			%	5	E	U	e	u			ง	ท	ศ	ๆ		◌
6			&	6	F	V	f	v			จ	ธ	ษ	ฯ	◌	◌
7			'	7	G	W	g	w			ฉ	น	ส	◌	◌	◌
8			(	8	H	X	h	x			ช	บ	ห	◌	◌	◌
9			)	9	I	Y	i	y			ซ	ป	ฬ	◌	◌	◌
A			*	:	J	Z	j	z			ฌ	ผ	อ	◌	◌	◌
B			+	;	K	[	k	{			ญ	ฝ	ฮ	◌	◌	◌
C			,	<	L	\	l	¦			ฎ	พ	ะ	◌	◌	◌
D			-	=	M	]	m	}			ฏ	ฟ		◌	◌	◌
E			.	>	N	^	n	∿			ฐ	ภ	า	◌	◌	◌
F			/	?	O	_	o				ฑ	ม	ำ	◌	◌	

รูปที่ 2 แสดงตารางรหัสภาษาไทยของม.เกษตรศาสตร์

เพื่อว่าจะทำให้ขนาดของฟังก์ชันกะทัดรัด อ่านได้ความง่าย โดยฟังก์ชันย่อย 1 ฟังก์ชัน จะทำหน้าที่ได้เพียงอย่างเดียวเท่านั้น ดังต่อไปนี้

1. Upcursor เป็นฟังก์ชันที่จะเลื่อนตำแหน่งของเคอร์เซอร์ให้เคลื่อนขึ้นไปอยู่ในบรรทัดถัดขึ้นไปจากตำแหน่งปัจจุบัน
2. Downcursor เป็นฟังก์ชันทำงานแบบเดียวกับ Upcursor( ) เพียงแต่จะเลื่อนถัดลงไป 1 บรรทัด
3. upputch เป็นฟังก์ชันแสดงผลตัวอักษรบนจอภาพ ในตำแหน่งที่อยู่แบบสระบนหรือวรรณยุกต์
4. Downputch เป็นฟังก์ชันแสดงผลตัวอักษรบนจอภาพ ในตำแหน่งที่อยู่แบบสระล่าง
5. delup เป็นฟังก์ชันที่ใช้เมื่อต้องการลบอักษรที่เป็นสระบนหรือวรรณยุกต์
6. deldown เป็นฟังก์ชันที่ใช้เมื่อต้องการลบอักษรที่เป็นสระล่าง
7. delnorm เป็นฟังก์ชันที่ใช้เมื่อต้องการลบอักษรที่เป็นแบบปกติ (ในบรรทัดปัจจุบัน)
8. istmiddle เป็นฟังก์ชันที่ return ค่าแบบ Boolean สำหรับอักษรในบรรทัดปกติ เช่น พยัญชนะ
9. istuper เป็นฟังก์ชันที่ return ค่าแบบ Boolean สำหรับอักษรสระบน
10. istlower เป็นฟังก์ชันที่ return ค่าแบบ Boolean สำหรับอักษรสระล่าง (สระอุ, สระอู)
11. iswanayuk เป็นฟังก์ชันที่ return ค่าแบบ Boolean สำหรับอักษรวรรณยุกต์

ฟังก์ชันหลัก tgetchar (บรรทัดที่ 63–73)

ฟังก์ชันนี้จะทำการรอรับอินพุทจากคีย์บอร์ดมาเก็บไว้ที่ตัวแปร ch โดยใช้ฟังก์ชัน-getch ซึ่งทำงานเหมือนคำสั่ง getchar ซึ่งเรามักใช้กันเป็นปกติ ต่างกันตรงที่จะไม่แสดงค่าที่ได้มีการแสดงค่าที่ได้บนจอภาพ หากค่าที่ได้เป็นค่า 'ESC' (0x1B) จะทำการเปลี่ยนค่าของ Thmode ซึ่งเป็นตัวแปรที่ใช้สำหรับการตรวจสอบว่า ขณะนี้กำลังใช้โหมดภาษาไทยหรืออังกฤษ โดยใช้ประโยคคำสั่งเงื่อนไข (conditional operator) สวิทช์

ที่ค่ากลับไปมาระหว่างค่า TRUE กับ FALSE ให้กับตัวแปร Thmode ลักษณะการใช้ประโยคแบบนี้ถือได้ว่าเป็นลักษณะเด่นของภาษาซี แทนการใช้ประโยค if ในลักษณะนี้ซึ่งจะยาวและไม่สวยงามดังนี้

```
if (Thmode = = TRUE)
    Thmode =  FALSE;
else Thmode =  TRUE;
```

การ return ค่าของฟังก์ชัน หากว่าอยู่โหมดปกติคือภาษาอังกฤษ ก็จะให้ค่านั้นกลับไปตามปกติ ยกเว้นเมื่อเป็นภาษาไทย (Thmode = TRUE) จะต้องนำค่า Ch ไปเทียบกับค่าตารางที่เก็บไว้ในอะเรย์ ของ Thaitable ซึ่งทำได้โดยใช้รูปแบบง่าย ๆ คือ Thaitable [Ch] เท่านั้น

ฟังก์ชันหลัก Tputchar (บรรทัดที่ 75–84)

ฟังก์ชันนี้เป็นเพียงการตรวจสอบค่า Ch ที่จะแสดงบนจอภาพ หากเป็นภาษาอังกฤษ (มีค่าน้อยกว่า 0x7E) หรือพยัญชนะ/สระหน้า/สระหลัง (มีค่าน้อยกว่า 0xDE) จะแสดงผลในบรรทัดปัจจุบัน (ใช้ฟังก์ชัน putch) หากเป็นรหัสของสระล่าง จะทำการเรียกฟังก์ชัน "Downputch" ส่งค่าในบรรทัดถัดไป ซึ่งตรงข้ามกับสระบน/วรรณยุกต์ จะเรียกใช้ฟังก์ชัน "Upputch" แทน

ฟังก์ชันหลัก tputs (บรรทัดที่ 86–93)

จะทำการพิมพ์ค่าอักษรที่อยู่ภายในสตริง S ใหปรากฏบนจอภาพ โดยใช้ฟังก์ชัน Tputchar ส่งอักษรขึ้นทีละตัว จนหมดโดยใช้ for-loop ตรวจสอบค่า "\0" ซึ่งเป็นเครื่องหมายบอกให้รู้ว่าได้ถึงจุดสิ้นสุดสตริงแล้ว

สำหรับอักษรภาษาไทย มีข้อพิเศษที่ควรสนใจ คือ กรณีที่มีสระบน (เช่น สระอิ, สระอี,....) และมีวรรณยุกต์เป็นตัวถัดมา เราจะส่งแบบปกติไม่ได้ เนื่องจากวรรณยุกต์ที่ส่งภายหลังนี้ จะปรากฏทับสระบนหายไป ดังนั้นจึงต้องทำการรวมรหัสอักษรสองตัวนี้และส่งเป็นรหัสอักษรใหม่ขึ้นแทน เช่น สระอิและตามด้วยวรรณยุกต์เอก จะต้องส่งรหัสอักษร " ิ่ " ไปปรากฏแทน เป็นต้น หน้าที่ในส่วนนี้ผมได้แยกออกมาเป็นฟังก์ชันย่อย "Codetch" แทน สำหรับเทคนิคการหารหัสอักษรใหม่นั้น ผมใช้วิธีการตรวจสอบค่าของสระบนก่อน เพื่อจะได้แยกใช้อะเรย์รหัสผสมที่เป็นชุดของสระบนนั้น ๆ ส่วนว่าจะเอาอะเรย์ตัวที่เท่าไรนั้น จะต้องนำเอาค่าวรรณยุกต์ตามหลังสระบนนั้นไปลบค่าของวรรณยุกต์เอกอีกครั้งหนึ่ง

ฟังก์ชันหลัก tgets (บรรทัดที่ 95–127)

ฟังก์ชันนี้จะรับค่าที่อินพุทมาจากคีย์บอร์ด ซึ่งมีความยาวได้สูงสุดไม่เกินค่า len ที่กำหนดไว้ ส่วนการทำงานของฟังก์ชันพอจะแยกได้สองส่วน คือ

1. กรณีรับข้อมูลมาเป็นค่า Backspace ซึ่งจุดประสงค์ของผู้ใช้ก็เพื่อต้องการจะลบค่าที่ได้อินพุทก่อนหน้านี้ การลบค่าในหน่วยความจำนั้นดูจะไม่ค่อยมีปัญหานัก เพราะเป็นเพียงการลดตำแหน่งของอะเรย์ลงหนึ่งตัวเท่านั้น และการลบบนจอภาพก็จะต้องถูกต้องด้วย กรณีพิเศษจะเกิดขึ้นคือถ้าบนจอภาพเป็นรหัสผสมของสระบนและวรรณยุกต์ หากลบวรรณยุกต์ไป ก็จะต้องส่งสระบนให้ปรากฏบนจอภาพอีกครั้งด้วย ตัวอย่างเช่น บนจอภาพเป็นรหัสอักษรผสม " ี่ " (0x F3) หากต้องการลบวรรณยุกต์จะต้องลบอักษร " ี่ " ซึ่งถือเป็นหนึ่งรหัสอักษรนี้ออกไป แล้วจึงค่อยส่งรหัสอักษร " ี " (0x E0) ให้ปรากฏบนจอภาพอีกครั้งหนึ่ง เป็นต้น หน้าที่นี้ได้แยกออกเป็นการทำงานของฟังก์ชันย่อย "delchar"

```
main ( )
{
  char string[25], Ch;
  tputs ("ใส่ข้อมูล:");
  Ch = tgetchar ( ) ;
  tputs ("โปรดเติมความ:");
  tgets (String) ;
{
```

รูปที่ 3 แสดงตัวอย่างโปรแกรมประยุกต์ใช้งาน

2. กรณีอินพุทปกติ หากเป็นภาษาอังกฤษจะไม่มีปัญหาเลย เพราะจะไม่มีกฎเกณฑ์ใด ๆ ทั้งสิ้น แต่สำหรับภาษาไทยแล้ว ดูจะมีรายละเอียดเพิ่มขึ้นอีก ซึ่งผมยึดถือตามลักษณะการเขียนภาษาไทย ดังนี้

2.1 อักษรเริ่มต้นของสตริง จะเป็นสระบนและวรรณยุกต์ไม่ได้

2.2 หากว่าได้อินพุทค่าของสระบน, วรรณยุกต์หรือสระล่างแล้ว จะอินพุทค่าเหล่านี้ซ้ำไม่ได้ เช่น ถ้าใส่สระอิแล้วจะใส่สระอิ, สระอี, สระอึ, สระออ ไม่ได้ เป็นต้น หน้าที่เหล่านี้ได้แยกออกเป็นฟังก์ชันย่อย "checkinp"

ทำความเข้าใจ

การอธิบายการทำงานของแต่ละฟังก์ชันโดยละเอียดนั้นทำได้ยาก ในส่วนตัวของผมแล้ว คิดว่าไม่มีวิธีใดที่จะทำให้เข้าใจโปรแกรมได้ดี เท่ากับการอ่านการทำงานของโปรแกรมจากตัวโปรแกรม (SOURCE CODE) จริง ๆ แต่การที่จะช่วยให้โปรแกรมสามารถง่ายต่อการทำความเข้าใจ ทั้งสำหรับตัวผู้เขียนและผู้ที่จะมาอ่านภายหลัง นอกจากจะใช้การตั้งชื่อต่าง ๆ ภายในโปรแกรมให้สื่อความหมายแล้ว การแยกออกเป็นฟังก์ชันย่อยนั้นก็เป็นสิ่งที่ดีเช่นกัน แต่ไม่ควรจะมีมากเกินไป เพราะจะเกิดความสับสนได้ง่าย

ส่วนการนำไปประยุกต์ใช้งานนั้น ได้แสดงให้เห็นถึงตัวอย่างโปรแกรมในรูปที่ 3 ซึ่งเป็นโปรแกรมสั้น ๆ เพียงการพิมพ์สตริงที่เป็นภาษาไทยบนจอภาพและรับข้อมูลเท่านั้น สำหรับการเขียนสตริงภาษาไทยในตัวโปรแกรมนั้น ผมใช้วิธีการโหลดโปรแกรม THAI.COM ซึ่งเป็นโปรแกรมที่ทำให้สามารถพิมพ์ภาษาไทยในแบบอักษรขอมได้ ลงในหน่วยความจำก่อน จากนั้นจึงใช้เอดิเตอร์ของเทอร์โบ หรือไซด์คิคสร้างโปรแกรมตามปกติ